Filomat 29:9 (2015), 1969–1981
DOI 10.2298/FIL1509969M

Published by Faculty of Sciences and Mathematics,
University of Niš, Serbia
Available at: http://www.pmf.ni.ac.rs/filomat

Hexagonal Arrays for Fault-Tolerant Matrix Multiplication

Emina I. Milovanović[a], Igor Ž. Milovanović[a], Mile K. Stojčev[a]

[a]*Faculty of Electronics Engineering, A. Medvedeva 14, 18000 Niš, Serbia*

Abstract. This paper describes mathematical procedure for designing hexagonal systolic arrays that implement fault-tolerant matrix multiplication. Fault-tolerance is achieved by introducing redundancy at algorithm level by defining three equivalent algorithms with disjoint index spaces. The essence of the proposed method is based on mapping data dependency graph that corresponds to the matrix multiplication algorithm, by an appropriate epimorphism, into a graph with desired properties. Since there is a 1:1 correspondence between the algorithm and it's graph representation, all transformations performed on the graph directly affect the algorithm. Chosen epimorphism depends on the projection direction vector $\vec{\mu} = [\mu_1 \ \mu_2 \ \mu_3]^T$ and enables obtaining hexagonal arrays with optimal number of processing elements (PEs) for the given matrix dimensions, which realizes fault-tolerant matrix multiplication for the shortest possible time for that number of PEs. The proposed procedure is formally described by explicit formulas and can be used as a software tool for automatic synthesis of fault-tolerant arrays.

1. Introduction

Matrix multiplication is one of the fundamental operation in many scientific and technical applications. It appears as a macro-function within many numerical and non-numerical problems. Typical examples include solving problems in linear algebra, such as finding inverse matrix, determinant, LU factorization, eigenvalues, etc., graph theory, such as transitive closure, finding all-pairs shortest paths, minimal spanning tree, etc. Matrix multiplication is very time consuming since it requires $O(n^3)$ computational steps. On the other hand, it is highly regular and posses inherent parallelism and is therefore suitable for implementation on parallel architectures such as processor or systolic arrays.

Nowadays, CMOS VLSI technology made it possible to develop massively parallel computational systems. However, shrinking dimensions of transistors, which bring lower operating voltages and smaller critical charges, make these devices prone to various kind of faults. These faults can be categorized into three types: transient, intermittent, and permanent faults. Transient faults are failures in which the circuit remains functional, but the value it generates is corrupted. Transient faults are caused by strikes from energetic particles, such as alpha particle strikes generated within the package of the chip, or neutron strikes caused by cosmic ray activity in outer space. Intermittent faults are caused by the combination of device aging and by stress created by environmental conditions during chip operation [1, 2]. Permanent errors can

2010 *Mathematics Subject Classification*. Primary 68M07; Secondary 68Q35
Keywords. matrix multiplication, fault-tolerance, systolic arrays.
Received: 15 September 2013; Accepted: 03 February 2014
Communicated by Dragan S. Djordjević
Research supported by Serbian Ministry of Education and Science, Grant No TR-32012 and TR-32009.
Email addresses: ema@elfak.ni.ac.rs (Emina I. Milovanović), igor@elfak.ni.ac.rs (Igor Ž. Milovanović), mile.stojcev@elfak.ni.ac.rs (Mile K. Stojčev)

occur because of manufacturing imprecision during chip fabrication. When a permanent fault is detected, the logic block must be isolated and disabled. Here we are interested in the toleration of transient and intermittent faults. To protect against these faults, some form of error detection and correction capabilities have to be involved. A variety of techniques have been proposed for handling transient and intermittent faults. Most of them are based on some kind of redundancy, ether information (such as ABFT), hardware (such as TMR), or time (such as RESO). In this proposal, fault-tolerance is achieved through triplicated computation followed by majority voting, but instead of using full hardware redundancy (i.e. triplication) to achieve fault-tolerance, we generate redundant results partially by means of hardware (by adding two extra columns of processing elements in the systolic array) and partially by means of time redundancy by slightly increasing computation time. We are interested in the designing of hexagonal systolic arrays with the optimal number of PEs with respect to the problem size which implement fault-tolerant matrix multiplication. In order to reduce the number of PEs and the execution time, we use the fact that during each computational step, matrix multiplication can be performed in any order without modifying the final result. By involving two additional columns of PEs and the reordering of computational steps, we achieve that our hexagonal arrays have $\Omega = N_3(\min\{N_1, N_2\} + 2)$ PEs and perform fault-tolerant matrix multiplication for $T_{\text{tot}} = 3\max\{N_1, N_2\} + \min\{N_1, N_2\} + N_3 - 2$ computational steps (N_1, N_2 and N_3 are dimensions of matrices, namely $\mathbf{A} = (a_{ik})_{N_1 \times N_3}$, $\mathbf{B} = (b_{kj})_{N_3 \times N_2}$ and $\mathbf{C} = (c_{ij})_{N_1 \times N_2}$).

The rest of the paper is organized as follows. Section 1 describes the problem of interest which will be considered. Mathematical background that serves as the basis for the synthesis of fault-tolerant systolic arrays is presented in Section 3. Section 4 deals with the procedure for the synthesis of systolic arrays with optimal number of processing elements. For that number of PEs the execution time is minimized. Section 5 concentrates on determining initial data arrangement that provides correct implementation of fault-tolerant matrix multiplication algorithm. Performance evaluation of the synthesized arrays which relates to the number of PEs and execution time are given in Section 6. Some concluding remarks are given in Section 7.

2. Problem Definition

Suppose $\mathbf{A} = (a_{ik})$ and $\mathbf{B} = (b_{kj})$ are rectangular matrices of order $N_1 \times N_3$ and $N_3 \times N_2$, respectively. Their product, $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$, $\mathbf{C} = (c_{ij})$, can be computed according to the following recurrent relations

$$\begin{aligned} c_{ij}^{(0)} &:= 0 \\ c_{ij}^{(k)} &:= c_{ij}^{(k-1)} + a_{ik}b_{kj}, \qquad k = 1, 2, \ldots, N_3 \\ c_{ij} &:= c_{ij}^{(N_3)} \end{aligned} \tag{1}$$

for $i = 1, 2, \ldots, N_1$ and $j = 1, 2, \ldots, N_2$.

The following systolic algorithm can be used to compute (1)

Algorithm_1
for $k := 1$ **to** N_3 **do**
for $j := 1$ **to** N_2 **do**
for $i := 1$ **to** N_1 **do**
$a(i, j, k) := a(i, j-1, k)$**;**
$b(i, j, k) := b(i-1, j, k)$**;**
$c(i, j, k) = c(i, j, k-1) + a(i, j, k) * b(i, j, k)$**;**

with the following identities

$$a(i, 0, k) \equiv a_{ik}, \quad b(0, j, k) \equiv b_{kj}, \quad c(i, j, k) \equiv c_{ij}^{(k)}, \quad c(i, j, 0) \equiv 0. \tag{2}$$

Note that computations in Algorithm_1, i.e. eqn. (1), with respect to indices i, j and k, are performed at basic permutations of the sets $\{1, 2, \ldots, N_1\}$, $\{1, 2, \ldots, N_2\}$, and $\{1, 2, \ldots, N_3\}$, respectively. However, this is

not necessary. The computations can be performed at arbitrary permutations of the aforementioned sets (see [8]). We call this property *permutability*.

Directed coordinated lattice graph $G = (P_{int} \cup P_{in}, D)$ can be joined to the Algorithm_1. This graph is placed in a three-dimensional Cartesian space generated by the unity vectors

$$\vec{e}_1 = [1\ 0\ 0]^T, \qquad \vec{e}_2 = [0\ 1\ 0]^T, \qquad \vec{e}_3 = [0\ 0\ 1]^T, \tag{3}$$

Set $P_{int} = \{\vec{p} = [i\ j\ k]^T\}$ defines vertices of graph G where active computations from Algorithm_1 are performed, while $P_{in} = P_{in}(a) \cup P_{in}(b) \cup P_{in}(c)$, where $P_{in}(a) = \{\vec{p}_a = [i\ 0\ k]^T\}$, $P_{in}(b) = \{\vec{p}_b = [0\ j\ k]^T\}$ and $P_{in}(c) = \{\vec{p}_c = [i\ j\ 0]^T\}$ are vertices where initial elements of matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C^{(0)}}$ are placed, respectively. Directed edges in graph G are defined by the column vectors of matrix $\mathbf{D}$

$$\mathbf{D} = \begin{bmatrix} \vec{e}_b^3 & \vec{e}_a^3 & \vec{e}_c^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

These vectors also define propagation direction of elements of matrices $\mathbf{B}$, $\mathbf{A}$ and $\mathbf{C}^{(0)}$, respectively.

Systolic array synthesis procedure, employed in this paper, is based on mapping the graph G on the plain orthogonal to the projection vector $\vec{\mu} = [\mu_1\ \mu_2\ \mu_3]^T$ (see [5]). By this mapping a directed pseudo-graph in two-dimensional Cartesian space is obtained. This graph corresponds to the desired SA. Here we are interested in the synthesis of two-dimensional hexagonal systolic arrays (2D HSA), suitable for fault-tolerant matrix multiplication. The first condition for SA to be suitable for fault-tolerant computations is that it realizes Algorithm_1 with pipeline period $\lambda \geq 3$. Pipeline period is defined as a time interval between two successive computations in a processing element (see [9, 11, 12]). Hexagonal array obtained according to the direction $\vec{\mu} = [1\ 1\ 1]^T$ satisfies this condition and it is well studied in the literature (see [7, 12]). Here we are interested in hexagonal arrays obtained for the directions $\vec{\mu} = \begin{bmatrix} 1 & 1 & -1 \end{bmatrix}^T$, $\vec{\mu} = \begin{bmatrix} 1 & -1 & 1 \end{bmatrix}^T$ and $\vec{\mu} = \begin{bmatrix} -1 & 1 & 1 \end{bmatrix}^T$. These arrays realize Algorithm_1 with pipeline period $\lambda = 1$, and therefore were not considered as suitable candidate architectures for achieving fault-tolerance. However, these arrays have shorter execution time of matrix multiplication algorithm than the array obtained by the projection vector $\vec{\mu} = [1\ 1\ 1]^T$. We are going to show that the fault-tolerant matrix multiplication can be efficiently implemented on these arrays as well, with shorter execution time than that of the one obtained by the projection vector $\vec{\mu} = [1\ 1\ 1]^T$.

In this proposal, without the loss of any generality, we are going to synthesize 2D HSA obtained by the projection direction vector $\vec{\mu} = [1\ 1\ -1]^T$, which implements fault-tolerant matrix multiplication algorithm. The other two arrays can be synthesized by the same methodology. Apart from being suitable for fault-tolerant computation, the synthesized array should have optimal (i.e. minimal) number of processing elements (PEs) for the given matrix dimensions. Also, the execution time of the array should be as small as possible for the given number of PEs. This requires substantial modification of the systolic array synthesis procedure described so far in the literature [3–7].

3. Mathematical Background

Results obtained in this section serve as a background for the synthesis procedure which will be described in the next one. We start our analysis from the directed graph $G = (P_{int} \cup P_{in}, \mathbf{D})$ which corresponds to the Algorithm_1. Depending on the relation between N_1 and N_2, we map vertices of graph G, defined by set P_{int}, either into set $P_{int}^{(0)}$ or $\tilde{P}_{int}^{(0)}$. This is performed by the isomorphisms φ_1 or φ_2, defined by the following equalities

$$\varphi_1\left(\vec{p} = \begin{bmatrix} i & j & k \end{bmatrix}^T\right) = \begin{bmatrix} 3i-2 & j & k \end{bmatrix}^T$$

and

$$\varphi_2\left(\vec{p} = \begin{bmatrix} i & j & k \end{bmatrix}^T\right) = \begin{bmatrix} i & 3j-2 & k \end{bmatrix}^T, \tag{5}$$

for each $\vec{p} \in P_{int}$. In this way we have obtained new directed graphs $G^{(0)} = (P_{int}^{(0)}, \mathbf{D})$ and $\tilde{G}^{(0)} = (\tilde{P}_{int}^{(0)}, \mathbf{D})$.

Now, for the projection vector $\vec{\mu} = [1\,1\,-1]^T$ we define two mappings $\mathbf{H}_1 = (\mathbf{M}_1, \vec{F}_1)$ and $\mathbf{H}_2 = (\mathbf{M}_2, \vec{F}_2)$ defined by

$$\mathbf{M}_1 = \begin{bmatrix} \vec{\mu} & \vec{e}_2 & \vec{e}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \qquad \vec{F}_1 = \begin{bmatrix} 0 \\ -1 \\ 3N_1 - 3 \end{bmatrix},$$

and (6)

$$\mathbf{M}_2 = \begin{bmatrix} \vec{e}_1 & \vec{\mu} & \vec{e}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}, \qquad \vec{F}_2 = \begin{bmatrix} -1 \\ 0 \\ 3N_2 - 3 \end{bmatrix}.$$

We use mappings $\mathbf{H}_1$ and $\mathbf{H}_2$ to obtain sets $\bar{P}_{int}^{(0)} = \{\vec{p}_1 = [u\,v\,w]^T\}$ and $\hat{P}_{int}^{(0)} = \{\vec{p}_2 = [u\,v\,w]^T\}$ from $P_{int}^{(0)}$ and $\tilde{P}_{int}^{(0)}$, respectively. These sets are obtained according to the following equalities

$$\begin{aligned} \vec{p}_1 &= \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{M}_1 \vec{p}^{\,(0)} + F_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3i-2 \\ j \\ k \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \\ 3N_1 - 3 \end{bmatrix} = \\ &= \begin{bmatrix} 3i-2 \\ 3i+j-3 \\ -3i+k+3N_1-1 \end{bmatrix} \end{aligned}$$

and (7)

$$\begin{aligned} \vec{p}_2 &= \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{M}_2 \vec{p}^{\,(0)} + F_2 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} i \\ 3j-2 \\ k \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \\ 3N_2 - 3 \end{bmatrix} = \\ &= \begin{bmatrix} i+3j-3 \\ 3j-2 \\ -3j+k+3N_2-1 \end{bmatrix}, \end{aligned}$$

for $i = 1, 2, \ldots, N_1$, $j = 1, 2, \ldots, N_2$, and $k = 1, 2, \ldots, N_3$.

Thus we have obtained two directed graphs $G_1^{(0)} = (\bar{P}_{int}^{(0)}, \mathbf{D})$ and $G_2^{(0)} = (\hat{P}_{int}^{(0)}, \mathbf{D})$ defined by the following index sets

$$\bar{P}_{int}^{(0)} = \left\{ \vec{p}_1 = \begin{bmatrix} 3i-2 & 3i+j-3 & -3i+k+3N_1-1 \end{bmatrix}^T \right\}$$

and

$$\hat{P}_{int}^{(0)} = \left\{ \vec{p}_2 = \begin{bmatrix} i+3j-3 & 3j-2 & -3j+k+3N_2-1 \end{bmatrix}^T \right\}.$$

Some important features of mappings $\mathbf{H}_1$ and $\mathbf{H}_2$ and corresponding directed graphs $G_1^{(0} = (\bar{P}_{int'}^{(0)}, \mathbf{D})$ and $G_2^{(0)} = (\hat{P}_{int}^{(0)}, \mathbf{D})$, are given in the subsequent theorems.

Theorem 3.1. *Matrices $\mathbf{M}_i$, $i = 1, 2$ are nonsingular and mappings $\mathbf{H}_i$, $i = 1, 2$, are injective.*

Proof The first part of the assertion of Theorem 3.1 directly follows from the equality $\det \mathbf{M}_i = \mu_i = 1 \neq 0$, $i = 1, 2$. The second part of the assertion will be proved by the contradiction on the example of mapping $\mathbf{H}_1$.

Suppose, contrary to the assertion of Theorem 3.1, that there are at least two different nodes of graph $G^{(0)}$ that are mapped by $\mathbf{H}_1$ into the same node of graph $\bar{G}^{(0)}$. Let $\vec{p}^{\,(1)} = [3i_1 - 2 \quad 3i_1 + j_1 - 3 \quad -3i_1 + k_1 + 3N_1 - 1]^T$ and $\vec{p}^{\,(2)} = [3i_2 - 2 \quad 3i_2 + j_2 - 3 \quad -3i_2 + k_2 + 3N_1 - 1]^T$, $\vec{p}^{\,(1)} \neq \vec{p}^{\,(2)}$, be position vectors of these nodes. In that case, according to (2), the following would be valid $\mathbf{M}_1 \vec{p}^{\,(1)} + \vec{F}_1 = \mathbf{M}_1 \vec{p}^{\,(2)} + \vec{F}_1$. This implies that

$$3i_1 - 2 = 3i_2 - 2, \tag{8}$$

$$3i_1 + j_1 - 3 = 3i_2 + j_2 - 3, \tag{9}$$

$$-3i_1 + k_1 + 3N_1 - 1 = -3i_2 + k_2 + 3N_1 - 1. \tag{10}$$

From (8) it follows $i_1 = i_2$. Accordingly, from (9) and (10) it follows $j_1 = j_2$ and $k_1 = k_2$, so we have $\vec{p}^{\,(1)} = \vec{p}^{\,(2)}$, which is in contradiction with the assumption that $\vec{p}^{\,(1)} \neq \vec{p}^{\,(2)}$.

□

On the basis of Theorem 3.1 we conclude that graphs $G^{(0)}$ and $G_1^{(0)}$ (i.e. $G^{(0)}$ and $G_2^{(0)}$) have equal number of nodes which are differently arranged in the coordinated space. Since $\mathbf{M}_1 \cdot \mathbf{D} = \mathbf{M}_1 \neq D$ (i.e $\mathbf{M}_2 \cdot \mathbf{D} = \mathbf{M}_2 \neq \mathbf{D}$), mappings $\mathbf{H}_1$ and $\mathbf{H}_2$ do not preserve adjacency of vertices from $G^{(0)}$.

Theorem 3.2. *Let l be an arbitrary line parallel to $\vec{\mu} = [1\ 1\ -1]^T$. If line l contains at least one point (vertex) of graph $G_1^{(0)}$ (i.e. $G_2^{(0)}$), then it contains exactly N_1, (i.e. N_2), vertices of this graph.*

Proof Without loss of generality we will conduct the proof for the graph $G_1^{(0)}$. Suppose $\vec{p}^{\,(1)} = \begin{bmatrix} [3i-2 & 3i+j_1-3 & -3i+k_1+3N_1-1 \end{bmatrix}^T$ and $\vec{p}^{\,(2)} = \begin{bmatrix} 3i+1 & 3i+j_1 & -3i+k_1+3N_1-4 \end{bmatrix}^T$ are position vectors of two vertices from graph $G_1^{(0)}$. Since $\vec{p}^{\,(2)} - \vec{p}^{\,(1)} = \begin{bmatrix} 3 & 3 & -3 \end{bmatrix}^T = 3\vec{\mu}$, we conclude that these vertices are placed on the line l parallel to the direction vector $\vec{\mu} = [1\ 1\ -1]$. By taking, one after the other, $i = 1, 2, \ldots, N_1 - 1$, we conclude that there are exactly N_1 vertices of $G_1^{(0)}$ on this line.

□

Corollary 3.3. *Let $L = \{l\}$ be a set of all lines that satisfy the conditions of Theorem 3.2. Then the intersection of these lines with some plain not parallel to l, has exactly N_2N_3 (i.e. N_1N_3) points.*

Corollary 3.4. *Let $L = \{l\}$ be a set of all lines that satisfy the conditions of Theorem 3.2. Depending on the relation between N_1 and N_2 and by the appropriate choice of mapping $\mathbf{H}_i$, $i = 1, 2$, it can be achieved that intersection of these lines with some plane not parallel to l, has exactly $N_3 \min\{N_1, N_2\}$ points.*

Note that in practice a plane that is orthogonal to line l is usually taken.

According to Theorem 3.2 and Corollaries 3.3 and 3.4 one can get a geometrical idea of mapping $G_1^{(0)}$ (i.e. $G_2^{(0)}$) into two-dimensional directed graph $\Gamma_1 = (V_1, \Delta_1)$ (i.e. $\Gamma_2 = (V_2, \Delta_2)$ which corresponds to the respective 2D SA. However we need explicit analytical procedure for obtaining graph Γ_1 (i.e. Γ_2). First we define epimorphism $\mathbf{S}$ which is joined to the projection vector $\vec{\mu} = [1\ 1\ -1]^T$. It maps three-dimensional directed graph $G_1^{(0)}$ (i.e. $G_2^{(0)}$) into two-dimensional directed graph $\Gamma_1 = (V_1, \Delta_1)$ (i.e. $\Gamma_2 = (V_2, \Delta_2)$. Epimorphism $\mathbf{S}$ is 2×3 matrix of the form

$$\mathbf{S} = \begin{bmatrix} \vec{S}_1^T \\ \vec{S}_2^T \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \end{bmatrix}, \tag{11}$$

where $\vec{S}_1^T$ and $\vec{S}_2^T$ are vectors defining a plane orthogonal to the projection vector $\vec{\mu} = [1\ 1\ -1]^T$. These vectors have to conform the following constraints [5]:

1. Since $\vec{\mu} = [1\ 1\ -1]^T$ is orthogonal to $\vec{S}_1^T$ and $\vec{S}_2^T$ it follows that $\vec{S}_1^T \cdot \vec{\mu} = \vec{S}_2^T \vec{\mu} = 0$, i.e.

$$s_{11} + s_{12} - s_{13} = 0 \qquad \text{and} \qquad s_{21} + s_{22} - s_{23} = 0. \tag{12}$$

2. In order to avoid mapping 3D graph $G_1^{(0)}$ (i.e. $G_2^{(0)}$) into 1D graph, the rank of matrix $\mathbf{S}$ must be 2, i.e. rank$\mathbf{S} = 2$.
3. To preserve the adjacency of nodes that were present in graph $G_1^{(0)}$, elements of matrix $\mathbf{S}$ can take values only from the set $\{-1, 0, 1\}$.

Conditions (1) to (3) do not necessarily provide one of the crucial requirements, which is that graph Γ_1 (i.e. Γ_2) has N_2N_3 (i.e. N_1N_3) vertices. The next theorem gives necessary and sufficient conditions for epimorphism $\mathbf{S}$ so that this requirement is always fulfilled.

Theorem 3.5. *Necessary and sufficient conditions that directed graph $\Gamma_1 = (V_1, \Delta_1)$ has $|V_1| = N_2N_3$ vertices are, apart from (1)-(3), the following ones*

$$\begin{aligned} &s_{12} = s_{23} = 0 \quad \textit{and} \quad s_{22}s_{13} \neq 0, \\ \textit{or} \\ &s_{22} = s_{13} = 0, \quad \textit{and} \quad s_{12}s_{23} \neq 0. \end{aligned} \tag{13}$$

Proof Let $\vec{p}^{\,(0)} = [3i - 2 \;\; j \;\; k]^T$ be a position vector of arbitrary point from $P_{int}^{(0)}$. Then we have that

$$(\mathbf{S} \cdot \mathbf{M}_1)\,\vec{p}^{\,(0)} = \begin{bmatrix} s_{12}j + s_{13}k \\ s_{22}j + s_{23}k \end{bmatrix}.$$

In order to have exactly N_2N_3 ordered pairs of the form $(s_{12}j + s_{13}k, s_{22}j + s_{23}k)$, necessary and sufficient conditions are the ones given by (13).

□

Similarly, the following theorem can be proved.

Theorem 3.6. *Necessary and sufficient conditions that directed graph $\Gamma_2 = (V_2, \Delta_2)$ has $|V_2| = N_1N_3$ vertices are, apart from (1)-(3), the following ones*

$$\begin{aligned} &s_{11} = s_{23} = 0 \quad \textit{and} \quad s_{21}s_{13} \neq 0, \\ \textit{or} \\ &s_{21} = s_{13} = 0, \quad \textit{and} \quad s_{11}s_{23} \neq 0. \end{aligned}$$

Note that epimorphism $\mathbf{S}$ can not be uniquely determined. For example, if $\mathbf{S} = [\vec{S}_1^T \;\; \vec{S}_2^T]^T$ is an epimorphism, so is $\bar{\mathbf{S}} = [\vec{S}_2^T \;\; \vec{S}_1^T]^T$. Therefore, without loss of generality, according to Theorems 3.5 and 3.6, and conditions (1)-(3), we can take the following epimorphisms

$$\mathbf{S}_1 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}, \qquad \mathbf{S}_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 1 & 0 \end{bmatrix} \tag{14}$$

respectively.

4. Systolic Array Synthesis

Based on the results obtained in the previous section, we are going to synthesize 2D HSA that realizes fault-tolerant matrix multiplication.

Since fault tolerance is achieved by triplicated computation of the same problem instance followed by majority voting, we first have to derive three matrix multiplication algorithms equivalent to Algorithm_1. To do that, we perform the shifting of vertices in the directed graph $G^{(0)} = (P_{int}^{(0)}, \mathbf{D})$ (i. e. $\tilde{G}^{(0)} = (\tilde{P}_{int}^{(0)}, \mathbf{D})$) along the coordinate axes to obtain two additional graphs $G^{(r)} = (P_{int}^{(r)}, \mathbf{D})$, (i.e. $\tilde{G}^{(r)} = (\tilde{P}_{int}^{(r)}, \mathbf{D})$, $r = 1, 2$ with the following set of vertices

$$P_{int}^{(r)} = \{\vec{p} = [3i - 2 - r \;\; j + r \;\; k]^T\} \quad \Big(\text{i.e.} \quad \tilde{P}_{int}^{(r)} = \{\vec{p} = [i + r \;\; 3j - r - 2 \;\; k]^T\}\Big)$$

Note that this shifting is not unique. The only thing we should take care of is that sets $P_{int}^{(r)}$, (i.e. $\tilde{P}_{int}^{(r)}$), $r = 0, 1, 2$ are mutually disjoint.

If we apply mapping $\mathbf{H}_1$ (i.e. $\mathbf{H}_2$), defined by (7), on the set of vertices of $G^{(r)}$ (i.e. $\tilde{G}^{(r)}$), $r = 1, 2$, we obtain directed graphs defined by the following set of vertices

$$\begin{aligned} &\bar{P}_{int}^{(r)} = \{\vec{p}_1 = [3i - 2 - r \;\; 3i + j - 3 \;\; -3i + k + r + 3N_1 - 1]^T\} \\ (\text{i.e.} \quad &\hat{P}_{int}^{(r)} = \{\vec{p}_2 = [3j + i - 3 \;\; 3j - 2 - r \;\; -3j + k + r + 3N_2 - 1]^T\}) \end{aligned}$$

for $r = 1, 2$.

Now we are able to define systolic algorithms that correspond to graphs $G_1^{(r)}$ (i.e. $G_2^{(r)}$), $r = 0, 1, 2$

Algorithm_2
for $r := 0$ **to** 2 **do**
for $k := 1$ **to** N_3 **do**
for $j := 1$ **to** N_2 **do**
for $i := 1$ **to** N_1 **do**
$u := 3i - 2 - r$;
$v := 3i + j - 3$;
$w := -3i + k + r + 3N_1 - 1$;
$a(u, v, w) := a(u, v - 1, w)$;
$b(u, v, w) := b(u - 1, v, w)$
$c(u, v, w) = c(u, v, w - 1) + a(u, v, w) * b(u, v, w)$;

i.e.

Algorithm_3
for $r := 0$ **to** 2 **do**
for $k := 1$ **to** N_3 **do**
for $j := 1$ **to** N_2 **do**
for $i := 1$ **to** N_1 **do**
$u := i + 3j - 3$;
$v := 3j - 2 - r$;
$w := -3j + k + r + 3N_2 - 1$;
$a(u, v, w) := a(u, v - 1, w)$;
$b(u, v, w) := b(u - 1, v, w)$
$c(u, v, w) = c(u, v, w - 1) + a(u, v, w) * b(u, v, w)$;

Denote with $\bar{\Gamma}_1 = (V_1^{(r)}, \mathbf{\Delta}_1)$ (i.e. $\hat{\Gamma}_2 = (V_2^{(r)}, \mathbf{\Delta}_2)$ image of $G_1^{(r)}$ (i.e. $G_2^{(r)}$), $r = 0, 1, 2$, obtained by epimorphism $\mathbf{S}_1$ (i.e. $\mathbf{S}_2$). This image is located in the 2D cartesian space and corresponds to the 2D HSA which implements the fault-tolerant matrix multiplication algorithm - Algorithm_2 (i.e. Algorithm_3). Denote the corresponding array by SA1 (i.e. SA2). Then (x, y) locations of the processing elements in the SA1 array (i.e vertices of Γ_1) are determined from the following equalities

$$PE \mapsto \begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{S}_1 \cdot \vec{p}_1 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 3i - 2 - r \\ 3i + j - 3 \\ -3i + k + r + 3N_1 - 1 \end{bmatrix} = \begin{bmatrix} k + 3N_1 - 3 \\ 1 - j - r \end{bmatrix}, \tag{15}$$

for $r = 0, 1, 2$, $j = 1, 2, \ldots, N_2$ and $k = 1, 2, \ldots, N_3$.

Directed edges in Γ_1, which correspond to communication channels in SA1, are determined by column vectors of matrix $\mathbf{\Delta}_1$

$$\mathbf{\Delta}_1 = \begin{bmatrix} \vec{e}_b^2 & \vec{e}_a^2 & \vec{e}_c^2 \end{bmatrix} = \mathbf{S}_1 \cdot \mathbf{D} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}. \tag{16}$$

Similarly, for the array SA2 we obtain that PE locations are determined by

$$PE \mapsto \begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{S}_2 \cdot \vec{p}_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 3j + i - 3 \\ 3j - 2 - r \\ -3j + k + r + 3N_2 - 1 \end{bmatrix} = \begin{bmatrix} k + 3N_2 - 3 \\ 1 - i - r \end{bmatrix}, \tag{17}$$

for $r = 0, 1, 2$, $i = 1, 2, \ldots, N_1$ and $k = 1, 2, \ldots, N_3$, and communication channels by the column vectors of matrix $\mathbf{\Delta}_2$

$$\mathbf{\Delta}_2 = \begin{bmatrix} \vec{e}_b^2 & \vec{e}_a^2 & \vec{e}_c^2 \end{bmatrix} = \mathbf{S}_2 \cdot \mathbf{D} = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 1 & 0 \end{bmatrix}. \tag{18}$$

5. Obtaining Initial Data Arrangement

So far we have obtained locations of the PEs in the x-y plane and directions of communication channels between them. Now we have to determine initial data arrangement that provides correct computation of the fault-tolerant matrix multiplication algorithm on the systolic array. For the sake of brevity we will do this only for the array SA1.

From Algorithm_2, i.e. corresponding directed graphs $G_1^{(r)}$, initial data arrangement of elements of matrices A, B and $C^{(0)}$ in a three-dimensional space is defined by the set $P_{in}^{(r)} = P_{in}^{(r)}(a) \cup P_{in}^{(r)}(b) \cup P_{in}^{(r)}(c)$, where

$$\begin{aligned} P_{in}^{(r)}(a) &= \{\vec{p}_a = \begin{bmatrix} 3i-2-r & 0 & -3i+k+r+3N_1-1 \end{bmatrix}^T\} \\ P_{in}^{(r)}(b) &= \{\vec{p}_b = \begin{bmatrix} 0 & 3i+j-3 & -3i+k+r+3N_1-1 \end{bmatrix}^T\} \\ P_{in}^{(r)}(c) &= \{\vec{p}_c = \begin{bmatrix} 3i-2-r & 3i+j-3 & 0 \end{bmatrix}^T\} \end{aligned}$$

for $r = 0, 1, 2$, $i = 1, 2, \ldots, N_1$, $j = 1, 2, \ldots, N_2$ and $k = 1, 2, \ldots, N_3$.

However this data arrangement does not provide correct execution of Algorithm_2 on the array SA1. This is illustrated in the Figure 1. To simplify, consider a directed graph with four vertices, P1 to P4, which belong to the internal computation space, P_{int}, where MAC (multiply with accumulation) operation is performed. With a, b, c, and d input data belonging to the initial computation space, P_{in} are denoted. For these inputs the following products ac, ad, bc and bd should be obtained. However, we will obtain only two products, namely ac and bd, since data are pipelined through the array (Fig. 1a)). Therefore we have to rearrange the space of initial data. In essence it is necessary to skew input data in time as illustrated in Fig. 1 b).

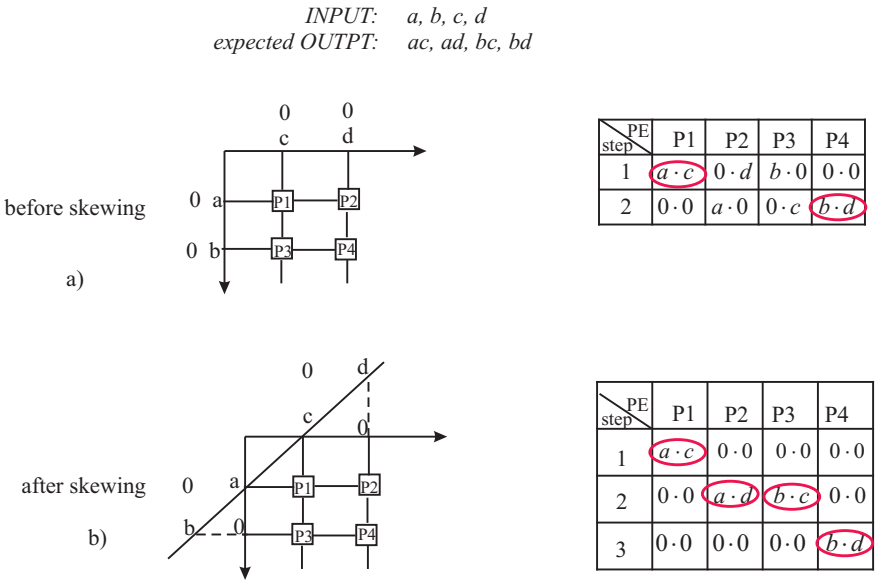

Figure 1: Illustration of computation in the array a) before skewing input data b) after skewing input data.

Skewing of $P_{in}^{(r)}$ is performed by the timing function, $t(\vec{p}) = \vec{\Pi}^T \vec{p} + \alpha$ (see [6]). This function defines time instance when the computation in the point $\vec{p} \in \bar{P}_{int}^{(r)}$ is performed. So we have to determine vector $\vec{\Pi}$ and constant α. Vector $\vec{\Pi} = [t_{11}\ t_{12}\ t_{13}]^T$ may be any vector orthogonal on the plane that contains all vertices of graph $G_1^{(r))}$, for some constant r, where computations from Algorithm_2 are performed at the same time instance. To narrow down choices, the following constraints for elements of vector $\vec{\Pi}$ are introduced:

- $|t_{11}| + |t_{12}| + |t_{13}| > 0$;
- The norm of $\vec{\Pi}$, $|\vec{\Pi}|^2 = |t_{11}|^2 + |t_{12}|^2 + |t_{13}|^2$, $t_{ij} \in Z$, should be as small as possible, and
- $\vec{\Pi}^T\vec{e}_b^3 > 0$, $\vec{\Pi}^T\vec{e}_a^3 > 0$, $\vec{\Pi}^T\vec{e}_c^3 > 0$, or $\vec{\Pi}^T\vec{e}_b^3 < 0$, $\vec{\Pi}^T\vec{e}_a^3 < 0$, $\vec{\Pi}^T\vec{e}_c^3 < 0$

Suppose that computation in Algorithm_2 is performed at the vertex $p \in P_{int}^{(r)}$, of graph $G_1^{(r)}$, for some constant r, with the position vector $\vec{p} = \begin{bmatrix} 3i-2-r & 3i+j-3 & -3i+k+r+3N_1-1 \end{bmatrix}^T$. Immediately prior to the computation at p, the computations in three points of $\bar{P}_{int}^{(r)}$ given by the following position vectors, $\vec{p}_{=}\vec{p} - \vec{e}_b^3$, $\vec{p}_{=}\vec{p} - \vec{e}_a^3$ and $\vec{p}_{=}\vec{p} - \vec{e}_c^3$, were performed. These points determine the plain, i.e. they are not collinear. Vector orthogonal to this plane is any vector parallel to vector

$$\vec{\Pi}^{*T} = (\vec{p}_2 - \vec{p}_1)^T \times (\vec{p}_3 - \vec{p}_1)^T = (\vec{e}_b^3 - \vec{e}_a^3)^T \times (\vec{e}_b^3 - \vec{e}_c^3)^T = \begin{vmatrix} i & j & k \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{vmatrix} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}.$$

Having in mind the above mentioned constraints for the elements of vector $\vec{\Pi}$, we can take $\vec{\Pi} = [1\ 1\ 1]^T$.

Constant α is determined by the appropriate choice of vertex in graph $G_1^{(r)}$ where the first computation, defined by Algorithm_2, should occur. Let it be a vertex defined by a position vector $\vec{p}_0 = [u\ v\ w]^T$, where $i = j = k = 1$. From the conditions $t(\vec{p}_0) = \vec{\Pi}^T\vec{p}_0 + \alpha = 0$ and $t(\vec{p}) > 0$, for each $\vec{p} \in \bar{P}_{int}^{(r)}$, $\vec{p} \neq \vec{p}_0$, we obtain $\alpha = -3N_1 + 1$, so we have that timing function is defined as

$$t(\vec{p}) = u + v + w - 3N_1 + 1.$$

Rearrangement (i.e. skewing) of $P_{in}^{(r)}$ is performed by the following mapping

$$\vec{p}_\gamma^* = \vec{p}_\gamma - (t(\vec{p}_\gamma) + 1)\vec{e}_\gamma^3, \qquad \gamma \in \{a, b, c\},$$

for each $\vec{p}_\gamma$ from $P_{in}^{(r)}$ and vectors $\vec{e}_\gamma^3$, $\gamma \in \{a, b, c\}$ defined by (4). The rearranged space of initial computations $\bar{P}_{in}^{(r)} = \bar{P}_{in}^{(r)}(a) \cup \bar{P}_{in}^{(r)}(b) \cup \bar{P}_{in}^{(r)}(c)$ is defined by

$$\begin{aligned}
\bar{P}_{in}^{(r)}(a) &= \{\vec{p}_a^* = \begin{bmatrix} 3i-r-2 & 1-k & -3i+k+r+3N_1-1 \end{bmatrix}^T\} \\
\bar{P}_{in}^{(r)}(b) &= \{\vec{p}_b^* = \begin{bmatrix} 2-j-k-r & 3i+j-3 & -3i+k+r+3N_1-1 \end{bmatrix}^T\} \\
\bar{P}_{in}^{(r)}(c) &= \{\vec{p}_c^* = \begin{bmatrix} 3i-r-2 & 3i+j-3 & 3+r+3N_1-6i-j \end{bmatrix}^T\}.
\end{aligned}$$

Finally, by mapping the set $\bar{P}_{in}^{(r)}$ using epimorphism **S** defined in (14), initial data arrangement of input elements **A**, **B** and $\mathbf{C}^{(0)}$ in the (x, y)-plane at the beginning of the computation in the array SA1 is obtained. The arrangement of input data in the (x, y)-plane is given by

$$\begin{aligned}
a(i, 0, -i+k+r+N_1-3) &\mapsto \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3i+k-r-3 \\ k+3N_1-3 \end{bmatrix} \\
b(0, i+j-1, -i+k+r+N_1-3) &\mapsto \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 5-3i-2j-k-r \\ 1-3i-j+3N_1 \end{bmatrix} \\
c(i, i+j-1, 0) &\mapsto \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1-r-j \\ 1-3i-j+3N_1 \end{bmatrix},
\end{aligned} \tag{19}$$

for $r = 0, 1, 2$, $i = 1, 2, \ldots, N_1$, $j = 1, 2, \ldots, N_2$ and $k = 1, 2, \ldots, N_3$. We assume that the following identities in (19) are valid: $a(i, 0, t + N_3) \equiv a(i, 0, t)$, $b(0, m + N_2, t + N_3) \equiv b(0, m, t)$, $c(i, m + N_2, 0) \equiv c(i, m, 0)$ for each $m = 1, 2, \ldots, N_2$ and $t = 1, 2, \ldots, N_3$. An example of the SA1 array that implements fault-tolerant matrix

algorithm for the case $N_1 = 4, N_2 = 3$ and $N_3 = 2$ is presented in Fig. 2. Dashed squares represent additional processing elements which represent hardware overhead due to fault-tolerance.

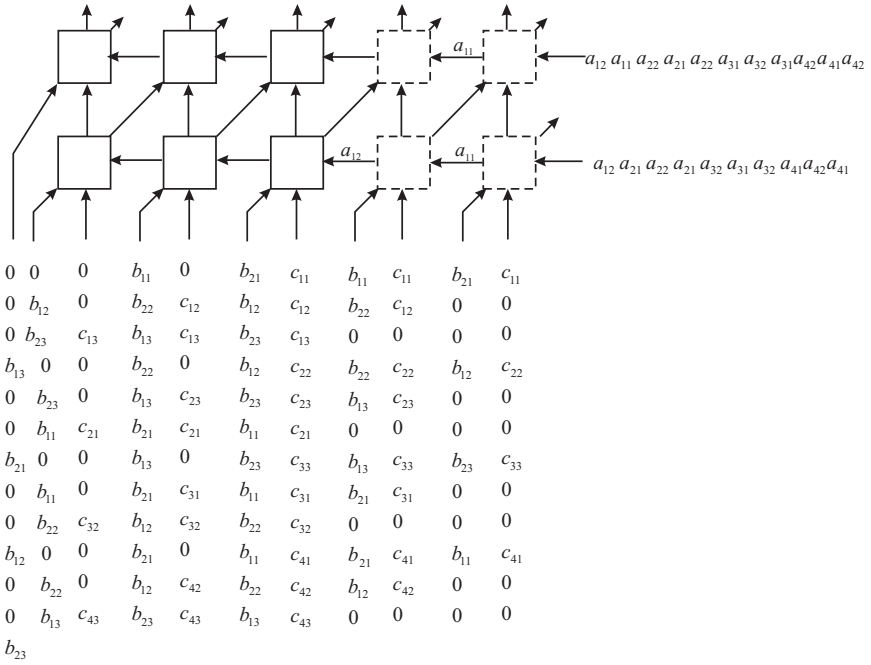

Figure 2: Fault-tolerant matrix multiplication on the array SA1 for the case $N_1 = 4, N_2 = 3$ and $N_3 = 2$.

By the similar procedure one can obtain initial data arrangement of input elements **A**, **B** and $\mathbf{C}^{(0)}$ in the (x, y)-plane at the beginning of the computation in the array SA2. It is given by

$$\begin{aligned} a(i+j-1,0,-j+k+r+N_2-3) &\mapsto \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 5-2i-3j-k-r \\ 1-i-3j-k+3N_2 \end{bmatrix} \\ b(0,j,-j+k+r+N_2-3) &\mapsto \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3j+k-r-3 \\ k+3N_2-3 \end{bmatrix} \\ c(i+j-1,j,0) &\mapsto \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1-i-r \\ 1-i-3j+3N_2 \end{bmatrix} \end{aligned} \tag{20}$$

for $r = 0, 1, 2$, $i = 1, 2, \ldots, N_1$, $j = 1, 2, \ldots, N_2$ and $k = 1, 2, \ldots, N_3$.

An example of the array SA2 that implements fault-tolerant matrix algorithm for the case $N_1 = 4, N_2 = 3$ and $N_3 = 2$ is presented in Fig. 2. As can be seen from Figs. 2 and 3 the array SA1 has fewer PEs than SA2 since $N_1 > N_2$.

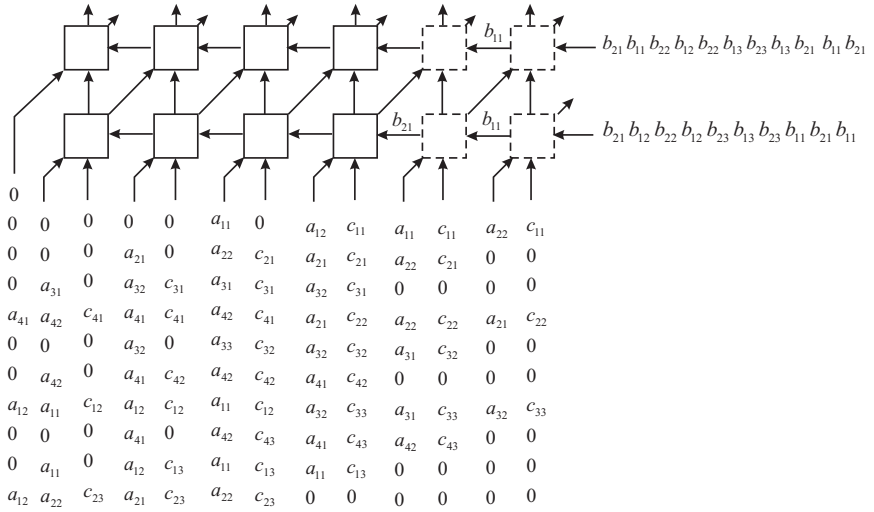

Figure 3: Fault-tolerant matrix multiplication on the array SA2 for the case $N_1 = 4, N_2 = 3$ and $N_3 = 2$.

A schematic of the voting logic is presented in Fig. 4. It consists of $3 * \left[\frac{\min\{N_1,N_2\}+2}{3}\right]$ three-input multiplexors and $\left[\frac{\min\{N_1,N_2\}}{3}\right]$ majority voters. Select signals for all multiplexors are common. The inputs are selected in a circular manner. In each cycle the majority voter takes three inputs and generates the result. More details concerning the majority voter can be found in [1].

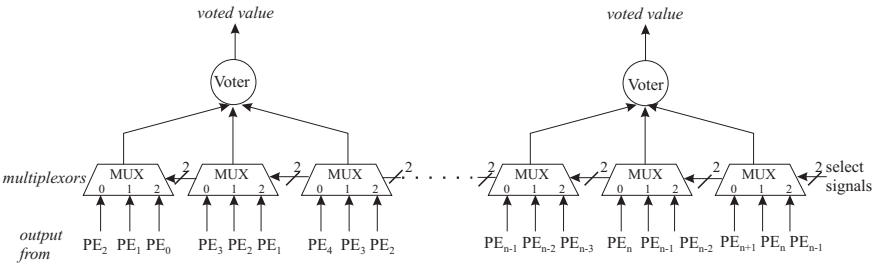

Figure 4: A detail of the voting logic.

By the proposed scheme single transient and intermittent faults can be corrected. A number of multiple fault patterns can be tolerated also, if they do not affect the same element of the resulting matrix. Fault detection and location are not necessary for fault-tolerance, errors are masked concurrently with normal operation of the systolic array.

6. Discussion

According to (15) and (17) the following result can be proved.

Theorem 6.1. *Hexagonal systolic array obtained by the projection vector* $\vec{\mu} = [1\ 1\ -1]^T$ *that implements Algorithm_2 (i.e. Algorithm_3) consists of*

$$\Omega = N_3(N_2 + 2) \qquad (\textit{ i.e. } \Omega = N_3(N_1 + 2)) \tag{21}$$

processing elements.

According to (21) it follows that optimal number of PEs in the SA depends on mutual relation between N_1 and N_2. This also means that this relation directly determines whether Algorithm_2 or Algorithm_3 should be used.

Corollary 6.2. *Hexagonal systolic array synthesized for projection vector $\vec{\mu} = [1\ 1\ -1]^T$ that implements Algorithm_2, i.e. Algorithm_3, has*

$$\Omega = N_3(\min\{N_1, N_2\} + 2) \tag{22}$$

processing elements.

Note that hexagonal SA synthesized for the direction $\vec{\mu} = [1\ 1\ 1]^T$ that implements fault-tolerant matrix multiplication obtained in [7] has the same number of PEs.

Total execution time, T_{tot}, of matrix multiplication algorithm on systolic array includes time to load data into the array, T_{in}, active execution time, T_{exe}, and time to drain the array, T_{out}, i.e. $T_{\text{tot}} = T_{\text{in}} + T_{\text{exe}} + T_{\text{out}}$. If $G = (P, E)$ is directed graph that corresponds to an algorithm, and $t(\vec{p})$ is timing function, then active execution time can be determined from the equation $T_{\text{exe}} = \max t(\vec{p}) - \min t(\vec{p}) + 1$, $\vec{p} \in \bar{P}^r_{int}$ [10]. In our case the following result holds.

Theorem 6.3. *Total execution time of Algorithm_2 (i.e. Algorithm_3) on the systolic array with optimal number of PEs obtained for the direction $\vec{\mu} = [1\ 1\ -1]^T$ is*

$$T_{\text{tot}} = 3\max\{N_1, N_2\} + \min\{N_1, N_2\} + N_3 - 2. \tag{23}$$

Proof Timing function of the array SA1 that implements Algorithm_2 is $t(\vec{p}) = u + v + w - 3N_1 + 1$. Consequently, active execution time is $T_{\text{exe}} = 3N_1 + N_2 + N_3 - 4$. In [4] hexagonal SA with optimal number of PEs which computes matrix product was synthesized for the direction $\vec{\mu} = [1\ 1\ -1]^T$. It consists of $N_3 \times N_2$ PEs, and has total execution time $T_{\text{tot}} = T_{\text{exe}}$, i.e. $T_{\text{in}} = T_{\text{out}} = 0$. The array SA1 synthesized in this paper represents an extension of the one obtained in [4] for $2N_3$ PEs. Namely, two PEs are added in each row of PEs. Since elements of matrix **A** propagate in that direction, the initialization time is increased for two time units, i.e. $T_{\text{in}} = 2$, while T_{out} remains 0. Accordingly, we have that total execution time of Algorithm_2 on the array SA1, synthesized by the equations (15), (16) and (19), is

$$T_{\text{tot}} = T_{\text{in}} + T_{\text{exe}} + T_{\text{out}} = 3N_1 + N_2 + N_3 - 2. \tag{24}$$

Similarly, it can be concluded that total execution time of Algorithm_3 on the array SA2, synthesized by equations (17), (18) and (20), is

$$T_{\text{tot}} = N_1 + 3N_2 + N_3 - 2. \tag{25}$$

Now, statement of the theorem directly follows from the equations (24) and (25).

□

Hexagonal array synthesized in [7] has total execution time equal to $T_{\text{tot}} = 3\max\{N_1, N_2\} + 2\min\{N_1, N_2\} + N_3 - 3$. This means that the array synthesized in this paper has shorter execution time for $\Delta T_{\text{tot}} = \min\{N_1, N_2\}$ time units which is achieved by the same number of PEs as in [7].

7. Conclusion

Systolic arrays are suitable accelerator architectures for matrix multiplication. They are complex CMOS VLSI circuits that exploit massive parallelism at data level. However, shrinking the dimensions of transistors make this circuits prone to various kind of faults. Reliable operation of these circuits can be achieved by some kind of redundancy, which in general involves additional hardware or time, that as a consequence

decreases system throughput. Therefore the main design challenges are oriented toward decreasing hardware complexity and increasing the throughput. In this paper we describe a systematic procedure for the designing of 2D hexagonal arrays that can tolerate single transient and intermittent faults during matrix multiplication (MM). In this proposal we start from the hexagonal array which implement MM algorithm with pipeline period $\lambda = 1$, and by involving appropriate modifications at algorithm level, obtain the array with $\lambda = 3$. Then two redundant computations are introduced so that fault-tolerance is achieved by triplicated computation followed by majority voting. This is achieved with minimal hardware and time overhead. The proposed procedure is formally described by explicit formulas and can be used as a software tool for automatic synthesis of fault-tolerant arrays.

References

[1] C. Constantinescu, Trends and challenges in VLSI circuit reliability, IEEE MIcro, Vol. 23, 4 (2003), pp. 14-19.

[2] M. Dubois, M. Annavaram, P. Stenstrom, Parallel computer organization and design, Cambridge University Prss, New York, 2012.

[3] J. A. B. Fortes, K. -S. Fu, B. W. Wah, Systematic design approaches for algorithmically specified systolic arrays, In: Computer Architecture: Concepts and Systems, (V. Milutinović, ed.), Elsevier Sci. Publ. Co., New York, 1988.

[4] I. Ž. Milovanović, M. P. Bekakos, I. N. Tselepis, E. I. Milovanović, Forty-three ways of systolic matrix multiplication, Inter. J. Comput. Math., Vol. 87, 6 (2010), pp. 1264-1276.

[5] S. G. Sedukhin, The designing and analysis of systolic algorithms and structures, Programming, Vol. 2, (1991), pp. 20-40. (In Russian)

[6] S. G. Sedukhin, G. Z. Karapetian, Design of optimal systolic systems for matrix multiplication of different structures, Report 85, Comput. Center Siberian Division of USSR Academy of Science, Novosibirsk, 1990. (In Russian).

[7] N. M. Stojanović, E. I. Milovanović, I. Stojmenović, I. Ž. Milovanović, T. I. Tokić, Mapping matrix multiplication algorithm onto fault-tolerant systolic array, Comput. Math. Appl., Vol. 48, 1-2 (2004), pp. 275-289.

[8] J. -C. Tsay, P. -Y. Chang, Some new designs of 2D array for matrix multiplication and transitive closure, IEEE Trans. Parall. Distrib. systems, Vol. 6, 4 (1995), pp. 351-362.

[9] C. R. Wan, D. J. Evans, Nineteen ways of systolic matrix multiplication, Inter. J. Comput. Math., Vol. 6, 1-2 (1998), pp. 39-69.

[10] J. H. Weston, C. N. Zhang, H. Li, Some space considerations on VLSI systolic array mappings, IEEE Trans. Circuits Systems II, Vol. 48, 4 (2000), pp. 419-424.

[11] C. N. Zhang, Systematic design of systolic arrays for computing multiple problem instance, Microelectronics J., Vol. 23, 7 (1992), pp. 543-553.

[12] C. N. Zhang, T. M. Bachtiar, W. K. Chou, Optimal fault-tolerant design approach for VLSI array processors, IEE Proc. Comput. Digit. Tech., Vol. 144, 1 (1997), pp. 15-21.

[13] C. N. Zhang, J. H. Weston, Y, -F. Yan, Determining objective functions in systolic array design, IEEE Trans. Very Large Scale Integration (VLSI) Systems, Vol. 2, 3 (1994), pp. 357-360.